Data Analysis Exercises - Day 2

Wouter Verkerke
(NIKHEF)

Exercise 5 – Unbinned Maximum Likelihood fit

- This a mostly a demonstration exercise only that shows some of the functionality and the syntax of the RooFit toolkit for data modeling that we'll be using in the next exercises
- Copy file ex5.C, look at it and run it. This macro does the following
  - It creates a Gaussian probability density function
  - It generates an unbinned dataset with 10k events from that function
  - It performs and unbinned ML fit of the Gaussian model to the toy dataset
  - It makes a plot of the data and overlays it with the Gaussian model
- Now comment out the part of the code labeled as 'block 1' and run the macro again
  - This code will print out the covariance matrix and correlation matrix of the fit parameters. Verify that cov(m,s)=corr(m,s)*σ(m)*σ(s) using the printed errors for mean.
- Now comment out the code labeled 'block 2' and run again.
  - This code will visualize the uncertainty of the model on the canvas using the error propagation technique of Ex 2. At 10K the event the uncertainty is very small (you can see it if you zoom in on the peak region of the pdf)

Exercise 5 – Unbinned Maximum Likelihood fit

  - Change the number of generated events from 10K to 100 and change the binning of the data in the plot from 100 bins to 10 bins (this is the argument in the w.var("x")->frame() call. Run again.
  - Lower the number of generated events from 100 to 10 and run again. The error on the shape will now be significant, but you see that an *unbinned* ML can reliably fit very small event samples
- Now comment out code block 3
  - This will visualize the error on the pdf shape due to the uncertainty on the mean parameter only

Exercise 6 – Analytical vs numeric MLE

- For certain simple cases it is possible to calculate the ML estimate of a parameter analytically rather than relying on a numeric estimate. A well known case is that of the fit of a lifetime of an exponential distribution
- Copy ex6.C and run it. This example performs an unbinned MLE fit to an exponential distribution of 100 events to estimate the lifetime.
- Now we aim construct the analytical estimator of the lifetime (do this part on paper, not by computer)
  - Write down the probability density function for the lifetime distribution.
  - It is essential that you formulate a normalized expression, i.e. Int F(t) dt == 1 when integrated from 0 to ∞. The easiest way to accomplish that is to divide whatever you expression you have by the integral of that expression over dt and then calculate that integral by hand

Exercise 6 – Analytical vs numeric MLE

  - Next write down the analytical form of the negative log-likelihood –log(L) for that probability density function given a dataset with values of x (label these x_i in your expression). **Be sure to also include the pdf normalization term in the expression**
  - The analytical ML estimate of tau then follows the requirement that d(-logL)/dtau = 0. Calculate the derivative of the –log(L) w.r.t. tau and solve the equation for the value of tau with results in the condition d(-logL)/dtau = 0

- Finally, implement the analytical calculation of MLE estimator for tau in the code of ex6.
  - Uncomment block one, which implements a look over the dataset, retrieving the values of the decay time one by one and build your calculation of the analytical estimate of tau with that of the numeric calculation from fitTo()
  - Explain why you might have minor discrepancies between the analytical and numeric calculations.
  - Increase the event count from 100 to 10000 and run again

Exercise 7 – Likelihood and correlations

- In this exercise we examine a model that intentionally introduces strong correlations and observe its effects
  - Copy ex7.C look at it and run it. The model that is being fit is the sum of two Gaussians with different (but similar) widths, a common mean and floating fraction between them.
  - Does the fit look OK to you (think about how you judge that)
  - Now uncomment BLOCK 1 and run again. This will visualize the error from the covariance matrix of the fit on the total pdf. Do the magnitude and shape of the uncertainty band look OK to you?
  - Now uncomment BLOCK 2 and run again. Examine the correlation matrix that is printed and look at the uncertainty of the background component. Do you understand the magnitude of the uncertainty of the background component vs that of the total distributions?
  - Now uncomment BLOCK 3 and run again. The correlation matrix shown in the previous exercise suggests that the chosen model has one almost redundant floating parameter. To mitigate this parameter sigma2 is fixed, is fitted again to data and the errors of the pdf and its background component are visualized again.

Exercise 7 – Likelihood and correlations

- Now we move to an examination of the –log(L) calculated from the model and the data
  - Uncomment BLOCK 4 and run again. You will now see a plot of –log(L) versus the fsig parameter. Does the –log(L) look parabolic to you?
  - Now zoom in on the area around the minimum (the range of fsig in which –logL rises by ~10 units w.r.t. 0. Does the likelihood look parabolic in this region.
  - Measure the interval of fsig in which –log(L) rises by 0.5 units. Compare this interval to the error reported on fsig by the fit and explain the difference.
  - Now uncomment BLOCK 5 and run again. The added code will draw contours in the likelihood function at –logL=+0.5 in (mean,fsig) and (sigma1,fsig). Do the shapes of the contours match your expectation from the correlation matrix obtained by the fit?
  - Now change the arguments “0.5,0,0” of both contour calls into “0.5,2,0” which will also draw the ‘2-sigma’ contours. How do you interpret the shape of these contours?
  - Finally change the arguments to “0.5,2,4.5” to also draw the ‘3-sigma contour and run again.

Exercise 8 – The effect of outliers on fits

- Outliers in distributions that are strongly peaked can create serious converges problems in fits that model such peaked distributions and do not take outliers into account.
  - Copy ex8.C, look at it and run it. This example generates a Gaussian distribution with a width that is relatively narrow compared to the defined range on x, and fits it to a Gaussian model.
  - Now uncomment BLOCK1. The added code 'manually' adds an event at x=3 and refits the distribution. Look at both fits carefully (just by eye, no need to make a chi2 check). Zoom in on the x axis if necessary. Does the outlier impact the result of the fit? (Also check the impact of the fitted value of mean,sigma)
  - Now move the position of the outlier event to x=4, run again and evaluate the situation again. Calculate what is the probability to obtain an event at x=4 or larger for this model? (You can use the 'TMath::Erfc' formula of Ex 1 to calculate this, but keep in mind that that formula evaluate the probability for |x|>Z, rather than x>Z)
  - Repeat the above exercise (including evaluation) at x=9.
  - Now uncomment BLOCK2. This fits the data to an improved model that foresees in a (small) flat background component that absorbs the outlier events and make the fit of the Gaussian component of the model function well in the presence of outliers. What value of fsig do you expect a priori to get out from the fit?
  - Now change the code fragment 'fsig[0,1]' by fsig[0.999] modifies to model to have a one permille fixed background component instead of a floating component. Rerun the fit. Does it still work well? Explain why (not)?

Exercise 9 – An MLE fit for an efficiency function

- This exercise demonstrates the procedure of an unbinned ML fit for an efficiency curve.
  - Copy ex9.C, look at it and run it. This macro create a data sample (x,c) in which c is a discrete observable that tells us if the event with value x has passed a selection (or not). The goal is to determine the efficiency eff(x) of the selection encoded in observable c.
  - The initial exercise creates an efficiency histogram from the dataset D(x,c) where each bin in x contains the fraction of events that have passed the cut. The efficiency histogram is constructed to have symmetric binomial errors on the data. Look at the slides of Module 1 to remind yourself how symmetric binomial errors are defined. An efficiency function 'fitfunc' is then fit to the data using χ^2 fit. Explain what approximation we are making by doing this?
  - Now uncomment BLOCK 1 of the the exercise and run again. This will make a plot 'all data' and 'accepted data', as well as perform an unbinned ML fit to measure the efficiency function. This fit is done using a *probability density function* F(c|x) that returns 'effFunc(x)' if the event is accepted and '1-effFunc(x)' if the event is rejected and is fit to the full dataset D(x,c)
  - Lower the number of events generated from 1000 to 200 (change variable N) and see what happens. Do the c2 and likelihood fits return correct results? Then lower N to 50 and run again.

Exercise 10 – Toy event generation

- This exercise demonstrates the principle of toy event generation through sampling.
  - Copy input file ex4.C and look at it. The input file defines a nearly empty main function and a function 'double func(double x)' is defined to return a Gaussian distribution in x.
  - The first step of this exercise is to sample func() to make a toy dataset. To do toy MC sampling we first need to know the maximum of the function. For now, we assume that we know that func(x) is a Gaussian and can determine the maximum by evaluating the function at x=0. Store the function value at x=0 into a double named fmax.
  - Now write a loop that runs 1000 times. In the loop, you generate two random numbers: a double x in the range [-10,10] and a double y in the range [0,fmax]. The value of x is a potential element of the toy dataset you are generating. If you accept it, depends on the value of y. Think about what the acceptance criterium should be (consult the slides of module 3 if necessary) and if it passes, store the value of x in the histogram.

Exercise 10 – Toy event generation

  - Allocate an integer counter to keep track of the number of accepted events. At the end of the macro draw the histogram and print the efficiency of the generation cycle, as calculated from the number of accepted events divided by the number of trial events
  - Now change the code such that instead of doing 10000 trials, the loop will only stop after 10000 *accepted* events. Modify the code such that you can still calculate the efficiency after this change.
  - Change the width of the Gaussian from 3.0 to 1.0. Run again and look at the generation efficiency. Now change it to 0.1 and observe the generation efficiency.

- Now we modify the toy generation macro so that it is usable on *any* function.
  - This means we can no longer rely on the assumption that the maximum of the function is at x=0.
  - The most common way to estimate the maximum of an unknown function is through random sampling. To that effect, add some code *before* the generation loop that samples the function at 100 random positions in x and saves the highest value found as fmax.

Exercise 10 – Toy event generation

- Change the width of the Gaussian back to 3.0 and run modified macro. Compare the fmax that was found through random sampling with the fmax obtained using the knowledge that the function was Gaussian (i.e fmax=func(0)).
- Now change the func(x) from the Gaussian function to the following expression:

  ```
  (1+0.9*sin(sqrt(x*x)))/(fabs(x)+0.1)
  ```

  and verify that it works fine.

- Finally we explore the limitations of sampling algorithms.
  - One runs generally into trouble if the empirical maximum finding algorithm does not find the true maximum. This is most likely to happen if you don't take enough samples or if the function is strongly peaked.
  - Choose the following func(x)

    ```
    TMath::Gaus(x,0,0.1,kTRUE)+0.1 ;
    ```

    i.e. a narrow Gaussian plus a flat background and rerun the exercise
  - Now lower the number of trial samples for maximum finding to 10 and see what happens

Exercise 11 – Chi-square on low statistics data

- This exercise explores the behavior of χ^2 on low statistics data
  - Copy ex5.C and look at it. This input file contains code that calculates the ingredients of a chi-squared (x,y,σy) given function func(x) and a ROOT histogram with data
  - The goal of this exercise is to generate a toy event sample from func(x) and then calculate the χ^2 of that dataset w.r.t. func(x).
  - The first step in this exercise is to add the 'toy' event generation code of the previous exercise to ex5.C which fills the histogram 'h' at the indicated location
  - Once the histogram is filled, complete the code that calculates the χ^2 from (data_x, data_y and data_ey). The macro is setup in such a way that exercise of generating toy data and calculating the χ^2 is repeated 1000 times and the resulting χ^2 values are stored in a histogram.
  - Examine the χ^2 distribution histogram. What is the number of degrees of freedom for this χ^2? Does it follow the expected χ^2 distribution.

Exercise 11 – Chi-square on low statistics data

- At very high nDOF the distribution of the χ^2 is asymptotically Gaussian. To determine to what extent this is the case, create an additional histogram with range [0,4] that you fill with χ^2/nDOF. Does it look approximately Gaussian?
- The issue is now that you do not know if the reduced χ^2 distribution is not Gaussian because one is not yet in the asymptotic regime, or of the input distribution is simply not consistent with the expected distribution. To make a better attempt at the latter we make an overlay of the χ^2 distribution with the expected distribution for the χ^2 for the number of degrees of freedom that have for this problem: The analytical form of the expected distribution of c2 for a given nDOF is present (but commented out) in the input file on the last line (the TF1 object).

  Uncomment the line, replace the tokens NEVT and NBIN and BINW (the width of each bin in units of x) with the appropriate values and add the line 'chi2f.DrawClone("same") ;'
  to overlay the expected distribution on the already drawn histogram. Is the data consistent with the expected distribution?

Exercise 11 – Chi-square on low statistics data

- To answer the above question more easily, it is customary to look at the distribution of the probability integral transform of the χ^2, rather than at the χ^2 itself (i.e. look at $\int_x^\infty P(\chi^2)d\chi^2$ instead of χ^2) The distribution if χ^2 value is consistent with the expected distribution, then the distribution of the probability integral transform values is expected to be flat. Create a histogram to hold the values of the probability integral transform and fill its value. Here you can use the function TMath::Prob(chi2,ndof) to calculate the probability integral transform value of chi2 for a given ndof. Is the distribution of Prob(chi2) approximately flat?
- What you see is that the examination of the prob(χ^2) distribution is a very sensitive test to test if a given distribution of χ^2 values follows the expected distribution.
- Increase the size of the sample from 200 to 1000 events and see if and how the behavior improves.
- Now switch from the flat distribution to a Gaussian distribution with mean 0 and width 5. Explain why switching from a uniform to a Gaussian distribution make the distribution look less like the expected distribution. Now lower the width of the Gaussian 2. What is the effect on this distribution?